International Journal of Computer Applications (0975 – 8887)
Volume 117 – No. 21, May 2015

Crowd-Enhanced Cloud Services: Issues and Directions

Saeed Arbabi
The University of Zabol
Zabol, Iran

Mohammad Allahbakhsh
The University of Zabol
Zabol, Iran

Mohsen Sharifi
Iran University of Science and Technology
Tehran, Iran

ABSTRACT

Cloud computing systems have emerged as a type of distributed systems in which a multitude of interconnected machines are gathered and recruited over the internet to help solve a computation or data-intensive problem. There are large numbers of cases in which Cloud techniques solely are not able to solve the job due to the nature of the tasks. To overcome this problem recently a strong inclination has emerged towards enlisting the human intelligence and wisdom of crowds a.k.a. Crowdsourcing in combination with the machine automated techniques. In this paper the authors propose a model for integrating crowds of people in the Cloud environments to enrich Cloud computing environments to be able to provide hybrid human-machine services enabling it to solve a wider variety of problems which some of them are studied here. The authors nickname these rich types of services, Crowd-enhanced Cloud services. At the end, the modality and challenges of this convergence and its future trends are explored.

General Terms

Cloud Computing, Crowdsourcing.

Keywords

Crowdsourcing, Cloud Services, Distributed Systems.

1. INTRODUCTION

Cloud computing systems are a type of distributed systems in which a collection of interconnected computing resources (e.g., CPU and storage) are shared, virtualized and presented as one or more unified computing services based on service-level agreements [1]. So cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service providers' interactions. Crowdsourcing also is an online distributed problem solving and production model that has emerged in recent years [2]. Crowdsourcing systems enlist a multitude of humans to help solve a wide variety of problems [3] ranging from labeling images [4], to finding 3-D protein structures [5], to classifying galaxies in Hubble images [6]. There are many problems for which one or more computational services in combination with human agents can actually offer whole or part of a badly needed service that can achieve a level of service quality that cannot be achieved under each regime solely. An example service is the human evaluation of the output of a language translation job done by machine; human query processing to complement database query processing [7]. Another example are big-data management approaches for large-scale classification of items, here the authors discuss Walmart product classification system [8] as a case-study in this paper.

This is the concept of "augmentation" that Doug Engelbart had proposed nearly 50 years ago in which machine-based services could improve semantically by integrating with human-provided services [9, 10]. Nowadays, crowds are increasingly employed in software and used alongside with the computers to solve problems. The same trend is emerging in the cloud services as well [11, 12]. Crowd-enhanced cloud which is the goal of this paper is a type of distributed system created as a result of injecting crowdsourcing into cloud computing at the lowest level of cloud hierarchy a.k.a resource pool. The authors aim to augment the cloud computing resource pool by considering the human crowd as a type of resource.

In this paper it is tried to find answers to the following questions in this paper:

1. Why is it necessary to enhance cloud services with a crowd (WHY)?
2. Where in the cloud hierarchy, the crowd should be injected (WHERE)?
3. What techniques exist for injecting a crowd into a cloud service (HOW)?

By answering these questions, the authors aim to define key research issues and articulate future research challenges and directions for crowd-enhanced cloud computing.

The rest of the paper is organized as follows. Section 2 explores the concept of crowdsourcing and some key issues in designing a crowdsourcing system. Section 3 defines cloud computing and exhibits several key characteristics of cloud computing systems. Section 4 discusses the relationship between Cloud computing and crowdsourcing in modern Web and then answers the three above questions; why, how and where to inject crowd into cloud. Next, the authors discuss issues and research challenges faced when injecting crowd into cloud in Section 6. Sections 7 and 8 conclude the work with a vision on future directions.

2. CLOUD SERVICE PRINCIPLES

In this section first an overview of the cloud computing service principles are studied and then some of the issues in this area are explored.

2.1 Overview

Clouds are vast pools of computing resources (e.g. processing, storage, applications) and cloud computing is a service-driven business model meaning that delivers these resources as a utility service via the internet on users' demands [13,14]. Although recently known as the revolution in IT industry [15], the idea behind the cloud computing, looked like a dream in 1960s when John McCarthy envisioned that the computing facilities will be provided to general public as a utility [8]. However, the "cloud computing" as a whole was first introduced in 2006 after Google's CEO Eric Schmidt coined the word to describe a business model for providing services across the Internet [16], and popularity of the word actually started sometime in October 2007 when IBM and Google announced a collaboration in that domain [17, 18]. Google, Amazon, Oracle Cloud, Rackspace, Salesforce, Zoho and Microsoft are some well-known cloud vendors.

Cloud computing vendors, a.k.a. cloud providers, provide their infrastructure (resource pool) as a utility to host service providers enabling them to provide their software services over Internet for service users. So service providers no longer need to be concerned about large capital outlays in hardware to host their services or experts to operate and maintain it [14, 19, 13]. Figure 1 shows an architecture for cloud computing. In cloud's service-driven business model, three general groups of services are provided: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

Infrastructure as a Service. Cloud providers usually virtualize infrastructure resources in an on-demand fashion for cloud infrastructure users. Amazon EC2 or the Rackspace Cloud are examples of IaaS cloud providers.

Platfrom as a Service. Cloud providers can provide as a service the software platform that systems run on. This software platform include operating system, software development framework and execution environments. Google App Engine, Microsoft's Azure cloud platforms are examples of well-known Platforms as Services.

Software as a Service. Cloud providers can provide applications over the Internet, giving users the chance to work with an application without locally installing it. Amazon Web Services is one example of Software as a Service.

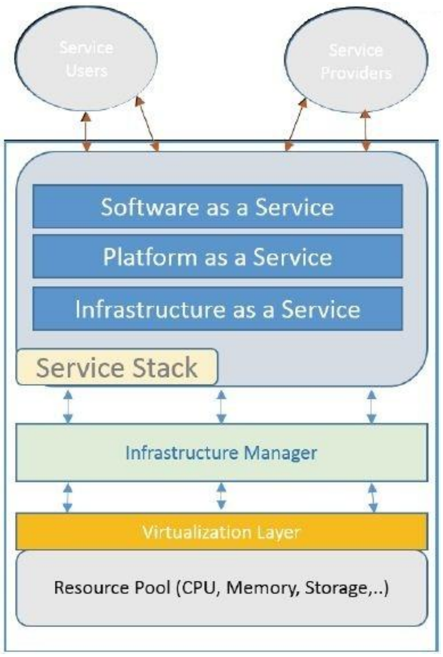

Fig 1: Cloud Computing Diagram

2.2 Dimensions

In this part some aspects of cloud computing model are explored.

Service Development. In cloud computing environments, cyber infrastructure developers develop and maintain the cloud framework. They are also in charge of hiding the infrastructure complexities from service users and higher level service providers.

Resource Provisioning. In modern clouds, services share the underlying computing resources by running in isolated virtual machines. Every cloud computing platform should be able to allocate and de-allocate resources on-demand to match the workloads on virtual machines. This is not easy to map service level objectives such as quality of service to low level hardware resources such as CPU or memory. There are two operations necessary for efficient resource provisioning: (1) static provisioning, when virtual machines are created with special size and then consolidated to a special set of physical machines [20], and (2) dynamic provisioning that allows virtual machine hardware resources to dynamically adjust themselves to match on-demand workloads [21, 22].

Quality of Service. The pool of computing resources is dynamically reconfigured and exploited based on the workloads on virtual machines running services. This is provided by the infrastructure provider with a guarantee on the performance and availability of service, a.k.a. Service Level Agreement (SLA) negotiated between the consumer and provider of a service. Due to dynamic nature of cloud resource allocation and de-allocation, continuous monitoring of quality of service is necessary to ensure SLA [14, 23]. Definition of the SLA with an appropriate level of granularity is an issue, so that it can completely cover the user needs and at the same time simple enough to be weighted, verified, evaluated and enforced by a resource provisioning mechanism in the cloud [24].

Cost and Payment Methods. Although moving onto cloud would considerably reduce the infrastructure costs, but at the same time it would raise the cost of data communication as the organization's data should transfer up to cloud and then down to organization. So organizations should consider this tradeoff when making decisions on migrating to the cloud [25]. From the point of view of a cloud provider, calculating the cost of consumed resources is a complicated task in dynamic resource provisioning cloud systems than static computing ones. Also a complete charging model needs to consider virtual machine associated items such as software licenses and virtual network usage in addition to above costs [24].

Service Interoperability. The concept of a cloud operated by one service provider or enterprise interoperating with a cloud operated by another is a powerful idea. Of course from within one cloud, explicit instructions can be issued over the Internet to another cloud. For example, code executing within Google AppEngine can also reference storage residing on AWS. However, there are no implicit ways that cloud resources and services can be exported or caused to interoperate.

3. CROWDSOURCING PRINCIPLES

In this section an overview of the crowdsourcing concept is provided and then the important dimensions of a crowdsourcing process are explored.

3.1 Overview

Crowdsourcing is the process of enlisting a crowd of people to solve a problem [26, 27]. The idea of crowdsourcing was first introduced by Jeff. Howe in 2006 [27]. Since then, an enormous amount of efforts has been put into this area from both academia and industry and many crowdsourcing platforms and research prototypes (either general or special purpose) have been introduced. Amazon Mechanical Turk1 (MTurk), Crowdflower2, Wikipedia3 and Stackoverflow4 are examples of well-known crowdsourcing platforms.

To crowdsource a problem, the problem owner, also called the requester, prepares a request for the crowd's contributions and submits it to a crowdsourcing platform. This request, also referred to as the crowdsourcing task or shortly as the task, consists of a description of the problem that is asked to be solved, a set of requirements necessary for task accomplishment, a possible criteria for evaluating quality of crowd contributions and any other information that can help workers produce contributions of higher quality levels. People who are willing to contribute to the task, also called workers, select the task if they are eligible to do so, and provide the requester with their contributions. The contributions are sent to the requester directly or through the crowdsourcing platform. The requester may evaluate the contributions and reward the workers whose contributions have been accepted [28, 29, 30].

3.2 Dimensions

Several dimensions characterize a crowdsourcing process, each of which impacts various aspects of the process from outcome quality to execution time and costs.

Task Definition. Task definition is important in the success of a crowdsourcing process. A poorly designed task can result in receiving low quality contributions, attracting malicious workers or leaving the task unsolved due to unnecessary complications [31, 32]. Therefore, it is highly recommended to design robust tasks. A robust task is designed so that it is easier to do it rather than to cheat [33]. Moreover, a requester should make sure to provide all information required by workers to do the task to increase the chance of receiving contributions of higher quality levels. The importance of this dimension is because of its direct impact mainly on the outcome quality, task execution time and the number of recruited workers.

Worker Selection. Quality of workers who contribute to a task can directly impact the quality of its outcome [33, 32, 34]. Low quality or malicious workers can produce low quality contributions and consequently waste the resources of the requester. Research shows that recruiting suitable workers can lead to receiving higher quality contributions [34, 35]. A suitable worker is a worker whose profile, history, experiences and expertise highly matches the requirements of a task. In a crowdsourcing process, workers might be recruited through various methods such as open-call, publish/subscribe [36], friend-based [37], profile-based [34] and team-based methods [38].

Real-time Control and Support. During the execution of the task, the requester may manually or automatically control the workflow of the task and manipulate the workflow or the list of the workers who are involved in the task in order to increase the chance of receiving high quality contributions [39, 40]. Moreover, workers may increase their experience while contributing to a task by receiving real-time feedback from other workers or requester. The feedback received in real-time, and before final submission of the worker's contribution, can assist the worker to pre-assess the provided contribution and enhance the contribution to satisfy the task requirements more fully [41]. Real-time workflow control and feedback can directly impact the outcome quality, the execution time and also the cost of the task, so they should be taken into account when studying crowdsourcing processes.

Quality Assessment. Assessing the quality of contributions received from the crowd is another important aspect of a crowdsourcing process. Quality in crowdsourcing is always under question. The reason is that workers in crowdsourcing systems have different levels of expertise and experiences; they contribute with different incentives and motivations; and even they might be included in collaborative unfair activities [33, 42, 43]. Several approaches are proposed to assess quality of workers' contributions such as expert review, input agreement, output agreement, majority consensus and ground truth [33].

Compensation Policy. Rewarding the workers whose contributions have been accepted, or punishing malicious or low quality workers, can directly impact their chance, eligibility and motivation to contribute to future tasks. Rewards can be monetary (extrinsic) or non-monetary (intrinsic). Research shows that the impact of intrinsic rewards, e.g., altruism or recognition in the community, on the quality of the workers' contributions is more than monetary rewards [44]. Choosing an adequate compensation policy can greatly impact the number of contributing workers as well as the quality of their contributions. Hence, compensation policy is an important aspect of a crowdsourcing process.

Aggregation Technique. A single crowdsourcing task might be assigned to several workers. The final outcome of such a task can be one or few individual contributions received from workers or an aggregation of all of them [23, 30]. Voting is an example of the tasks that crowd contributions are aggregated to build up the final task outcome. In contrast, in competition tasks only one or few workers' contributions are accepted and rewarded. Each of the individual contributions has its own characteristics such as quality level, worker's reputation and expertise. Therefore, combining or even comparing these contributions is a challenging task and choosing a wrong aggregation method can directly impact the quality of the task outcome.

Integration. A drawback of existing crowdsourcing platforms is that each defines a stand-alone system with rigid structure and requirements, and thus demands significant work in order to integrate human computation into larger applications. Each new application may require building a pipeline from the ground up, and in many cases, a new community. Particularly for complex applications, which may involve several steps of human computation using different crowdsourcing platforms interleaved with machine computation, constructing such a pipeline can be a tedious effort. In practice, complex systems are discouraged, and most uses of human computation avoid multiple interleaved processing steps.

4. INJECTING CROWD INTO CLOUD SERVICES

Nowadays, crowd are increasingly employed in software and used alongside with computers to solve problems. The same trend is emerging in the cloud services as well [11,12]. Therefore, in this section it is tried to present justifications on cloud-crowd computing by answering the following questions:

1. Why is it necessary to enhance cloud services with a crowd (WHY)?
2. In which level or levels of the cloud service stack (Fig.1), a crowd should be injected (WHERE)?
3. What techniques exist for injecting crowd sources into cloud (HOW)?

4.1 Why to inject Crowd into Cloud?

To answer this question, one should recall the definition of a cloud service presented in Section 3. According to this definition, one of the main resource categories that a cloud service can offer to its users is the computing resource. This fact is also depicted in Figure 1. For example, CPUs as computing resources can be granted and revoked to users based on their requests and requirements.

On the other hand, the most important aspect of crowdsourcing that differentiates it from other distributed systems is harnessing crowd intelligence and wisdom to solve a problem. In other words, crowdsourcing services employ people as computing resources to solve a certain task. People as computing resources can be involved in any type of task, but their major strength points is that they perform well in tasks that are pretty hard for computers but very easy for humans. Photo tagging is a very good example of these type of tasks. These tasks are also called Human Intelligence Tasks (HITs) [45]. Due to the rapid application of computers and mobile devices in people's daily life and the emerging field of mobile cloud computing platforms, a very wide variety of problems, a reasonable amount of them HITs, show up in our daily life. Probably, the best way to tackle these problems, mainly HITs, is to use crowdsourcing techniques in the cloud service context. Therefore, it is vital to enhance the cloud services with injecting the crowd computing power into cloud.

4.2 Where to inject Crowd?

Computing resources in clouds are managed in the lowest level of the hierarchy, a.k.a. infrastructure level or resource pool. Therefore, the authors here aim to inject crowd of people as a resource in the resource pool creating a rich type of cloud. Such augmentation enriches all service levels of target cloud computing system without changing the stack.

There are some crowdsourcing systems that have tried to inject crowd into cloud. An example is the crowdsourcing system reported in [12] that considers a crowdsourcing system with the mission of video quality testing and then tries shifting the videos onto the cloud, thus optimizing the production of videos to the workers. Lenk et al based on such works, redesigned the cloud computing service stack and offered a new crowdsourcing layer on top of it entitled human as a service [46].

Hybrid cloud architecture, presented in [47] also includes human as a service layer. This is the most similar work to our research, but it changes the cloud stack and their view is limited to software as a service layer while this work's contribution embeds crowds as resources into the cloud computing original stack without changing it and trying to have resource type transparency in mind. There exists some attempts at injecting crowd into cloud. An example is the work reported in [12].

4.3 How to inject the Crowd?

In [48] inevitable change in data-centric systems is considered as an effect of connectivity of billions of device-enabled people to the massive cloud computing infrastructure and is envisioned by developing hybrid cloud/crowd systems but no clear solution is suggested.

Crowd-servicing is the name picked for evolving vision in Web 3.0 by [7] in which services provided by human agents and machines are integrated over the World Wide Web to represent the full flowering of the augmentation concept [9].

Combining the power of both machine and human clouds within a hybrid cloud/crowd design that combines query Web services, string similarity algorithms and crowdsourcing in a real project case. The goal of this research as a proof of concept was to show that crowdsourcing can be used effectively and efficiently as part of software engineering practices [11].

Some works specially the one reported in [49] tried empowering the crowdsourcing service by changing it to be cloud-enabled. Delivering a general purpose crowdsourcing capability is studied in some works. The work done in [49] explores the technical requirements for delivering an end-to-end general purpose crowdsourcing platform in four sections: registration, initializing crowdsourcing request, carrying out crowdsourcing request, and complete crowdsourcing request. Then it has studied the extent to which four different crowdsourcing platforms meet the specific subset of these technical requirements. She demonstrates how the cloud infrastructure can be used as a scalable hosting and application development environment for dynamic, task-based crowd teaming.

A step forward in building a general purpose crowdsourcing platform and creating a tool for crowdsourcing programmers is reported in [28]. They have created a human and machine resource management system called Dormouse, and explore the drawbacks of current crowdsourcing paradigms such as treating human workers as homogenous and rigid structure that makes them hard to integrate with other platforms. Other human computation frameworks have been presented in [50, 51, 52] but they are all platform-dependent design patterns compared to jabberwocky, which is a full stack general purpose crowdsourcing.

Another type of effort for a crowdsourcing toolkit is usage of declarative query languages for human computation introduced in [53, 54]. The concept of global brain [55] is described to view all the people and computers on our planet. Because of the involvement of people, the programming skills needed for global brain is completely different from traditional computer programming and software engineering. Then the challenge of programming this global brain is studied.

There are real-time or single-worker applications, for example the work presented in [56], which is a computer-implemented method and system (Legion) for capturing and outsourcing an existing graphical user interface (GUI) of an application to a crowd for their collaborative real-time control using an input device.

5. CROWD-ENHANCED CLOUD SERVICE STACK

As discussed in Section 4, one should consider both people and machines as first-class citizens in cloud. Such injection empowers all the above layers of the cloud service stack, including Iaas, Paas and SaaS. In this section we clarify this consideration.

5.1 Crowd-Enhanced IaaS

A human and machine resource management system is needed on top of resource pool that feeds IaaS layer directly, like the virtualization layer in every other cloud service stack shown in Fig 1. This layer should interact with both people and machines, and in addition maintain real identities, user profiles, and social relationships for the people who comprise the system, and allow IaaS provider to define arbitrary person-level properties and social structures in the system. Further, because this virtualization layer defines communication protocols for both people and machines, by means of that, IaaS users could naturally interact with both in unified control flows even for complex parallel processing tasks. An example of such resource management layer named Dormous is presented in [28].

In existing crowdsourcing systems, human workers are often treated as homogeneous and interchangeable, which is useful in handling issues of scale and availability. However, limited notions of identity, reputation, expertise, and social relationships limit the scope of tasks that can be addressed by these systems. Incorporating real identities, social structure, and expertise modeling has proven valuable in a range of applications, for example, in question-answering with Aardvark. Building general frameworks for human computation that include these notions will enable complex applications to be built more easily.

5.2 Crowd-Enhanced PaaS

Some large-scale data applications such as MapReduce platform can be proposed as PaaS in crowd-enhanced cloud on top of IaaS. This MapReduce implementation can facilitate complex data processing tasks. Like main MapReduce, it gives the PaaS user the ability to specify map and reduce steps, but allowing either step to be powered by human or machine computation. The data flow, resource allocation, and parallelization necessary for each step are handled by this Crowd-Enhanced MapReduce with no onus on the PaaS user.

In addition to combining machine and human computation, crowd-enhanced MapReduce also provides the ability to choose particular types of people to complete each task (based on crowd-enhanced IaaS), and allows arbitrary dependencies between multiple map and reduce steps. Many interesting social computing applications fit naturally into this paradigm, as they frequently involve the need for parallelization of subtasks across people or machines, and subsequent aggregation such as writing a summary or averaging the ratings. As a simple example, conducting a survey and tabulating summary statistics for each question (breaking down according to a variety of demographics) can be expressed using a human map step that sends the survey in parallel to many people, and one or more machines reduce steps on the output that aggregate the responses keyed by question and/or user demographic. One such implemented platform named ManReduce is reported in [28].

5.3 Crowd-Enhanced SaaS

Hybrid cloud architectures, presented in [47] and [46] also include human as a service layer on top of main SaaS. In this work the authors don’t change the cloud main stack but enrich it with the human in resource pool, so here the authors don’t propose a human as a service layer but a crowd-enhanced software as a service layer. A drawback of existing crowdsourcing platforms is that each defines a stand-alone system with rigid structure and requirements, and thus demands significant work in order to integrate human computation into larger applications. Each new application may require building a pipeline from the ground up, and in many cases, a new community. Particularly for complex applications, which may involve several steps of human computation using different crowdsourcing platforms interleaved with machine computation, constructing such a pipeline can be a tedious effort. In practice, complex systems are discouraged, and most uses of human computation avoid multiple interleaved processing steps.

Most of the works done in the crowdsourcing to combine pure human-enabled Web services with machine-enabled Web services, could fit in this layer. The research done in [7] introduces the concept of crowdservicing. It then uses a scenario to illustrate how a crowdsourcing application as a service can provide all or part of a badly needed service, which, in combination with one or more computational services, can achieve a level of quality that each regime can’t achieve separately.

The work done in [12] considers a crowdsourcing system with the mission of video quality testing and then tries shifting the videos onto the cloud, thus optimizing the production of videos to the workers. So providing such crowdsourcing systems as services in SaaS layer can derive a benefit from cloud storage, which is the aim of that work.

5.4 Case-Study: Wall-Mart Product Classification

Wall-Mart product classification project is an example of hybrid crowd-machine approaches which is proposed for big data analytics [8]. In this project, a huge volume of data is constantly being received from various retailers from all across the country. The sent data is structured but in most of the cases it is incomplete. So, Wall-Mart cannot use only machines for the purpose of entity matching and resolution. The results, as tested, do not have the required level of accuracy. To overcome this problem, Wall-Mart selects some individuals with adequate expertise to refine the results produced by machine.

6. RESEARCH ISSUES

Injecting crowds of workers as a computing resource in cloud resource pool raises design challenges. In this section, the authors explain what issues raise when one injects crowd into cloud.

6.1 Large-Scale Data Management

Parallel programming frameworks, including MapReduce, Dryad [52], and GPU shader languages [3] have been developed in an environment characterized by data-intensive applications and the availability of parallel computing resources. This characterization also holds for many applications in human computation, for example, in image processing or machine learning. There are some applications where embedding crowd into MapReduce is not suitable, for example, for using a single worker in the crowd to control a robot [56]. So data-processing applications are well-suited to this approach, while this is not true for real-time or single-worker sequential applications.

6.2 Human and Machine Resource Management System

The need for a "virtual machine” layer in stack that consists of low-level software libraries is a research issue. This layer should interact with both people and traditional computing machines and maintain real identities, rich user profiles, and social relationships for the people who comprise the system, and allow end users to define arbitrary person-level properties and social structures in the system. Further, because this virtualization layer defines communication protocols for both people and machines, by means of that, programmers could very naturally interact with both in unified control flows even for complex parallel processing tasks. An example of such resource management layer is created in [28] named Dormous.

6.3 Supporting Social Computing

The other important aspect is to allow the programmer to route tasks based on personal properties such as expertise and demographic. For example, specify that certain tasks be dispatched only to people with graduate degrees in biology, or expertise in computer science, or simply to people under 25 or the people of one college.

6.4 Simple Task/Service Definistion

The crowd-enhanced cloud system should define simple interfaces for using, creating, and sharing functions (human or machine) and micro-task templates, making it easy to quickly implement a wide range of applications. This means for crowd-enabled tasks the need for straight forward mechanisms to create new templates for human tasks, and importantly, to reuse those created by others. This minimizes redundant work and allows programmers to focus on control flows rather than creating and optimizing task templates.

6.5 Quality Assessment

There are mechanisms for quality control and assessment of service both in cloud computing and crowdsourcing environments. In a crowd-enhanced cloud computing system a quality control mechanism is necessary to ensure some service level agreements in the parts of the service that use both human or machine resources.

6.6 Interoperability

Integration of computational and human-assisted services in two crowd-enhanced clouds is a design issue needed to be explored. For example, a service in one crowd-enhanced cloud, in one step, routes tasks to a large number of inexpensive workers from one crowd-enhanced cloud computing platform, and in a next step, routes tasks to a smaller number of vetted experts from another platform, without requiring to learn the separate (and often complex) API calls from multiple platforms.

6.7 Dynamic Pricing Mechanism

A dynamic pricing component is needed. Software components can be developed, which will adjust the pricing of crowdsourcing requests according to demand and participant availability alongside machine resource providing. The pricing mechanism should be something like pay as you go in clouds

7. CONCLUSION AND FUTURE DIRECTIONS

Utilizing the map-reduce could be a future direction. For example, machine map step could specify automatic information extraction of facts from a set of scientific papers, such as genetic and environmental risk factors for a set of candidate diseases. Then, a human reduce step could aggregate the proposed facts about each disease, check their accuracy and convert them into a summary.

The vast potential one can realize by integrating computation with human intellectual power has been recognized from computing earliest days. Its origins are in cybernetics, in the seminal writings of Norbert Wiener, Hal Ashby, Lick Licklider, and others in the 1950s and 60s. Wiener advocated for human use of human beings [57], whereas Ashby developed the concept of synthetically amplifying human intellectual power, which he referred to as intelligence amplification [58]. Licklider argued passionately for tightly coupling human intellect and computing machines to achieve man computer symbiosis. [59] He conjectured that the resulting systems can potentially perform at a level superior to each subsystem. In many situations, human agents can provide all or part of a badly needed service, which, in combination with one or more computational services, can achieve a level of quality that each regime cannot achieve separately. This is the potential that having crowdsourcing as services offers.

The existing crowdsourcing systems are often purpose-built, supporting a set of specific, micro tasks in a particular domain and a specific part of the product life-cycle. To address the identified gaps in architectural support for building crowdsourcing service it could embark on building a platform for crowdsourcing, demonstrating how the cloud infrastructure can be used as a scalable hosting and application development environment for dynamic, task-based crowd teaming.

In this paper the authors demonstrated why, how and where to inject crowd into cloud and then analyzed some research issues and challenges that arise from this convergence

8. REFERENCE

[1] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utili- ties. In High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on, pages 5–13. Ieee, 2008.

[2] Daren C Brabham. Crowdsourcing as a model for problem solving an introduc- tion and cases. Convergence: the international journal of research into new media technologies,14(1):75–90,2008.

[3] Anhai Doan, Raghu Ramakrishnan, and Alon Y Halevy. Crowdsourcing systems on the world-wide web. Communications of the ACM, 54(4):86–96, 2011.

[4] Luis Von Ahn. Games with a purpose. Computer, 39(6):92–94, 2006.

[5] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popovi´c, et al. Predicting protein structures with a multiplayer online game. Nature, 466(7307):756–760, 2010.

[6] Chris J Lintott, Kevin Schawinski, Anˇze Slosar, Kate Land, Steven Bamford, Daniel Thomas, M Jordan Raddick, Robert C Nichol, Alex Szalay, Dan Andreescu, et al. Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. Monthly Notices of the Royal Astronomical Society, 389(3):1179–1189, 2008.

[7] Joseph G Davis. From crowdsourcing to crowdservicing. Internet Computing, IEEE, 15(3):92–94, 2011.

[8] Douglas F Parkhill. Challenge of the computer utility, 1996.

[9] Douglas C Engelbart. Augmenting human intellect: Experiments, concepts, and possibilities. Technical report, DTIC Document, 1965.

[10] Joseph Davis and Huairen Lin. Web 3.0 and crowdservicing. 2011.

[11] Deniz Iren, Gokhan Kul, and Semih Bilgen. Utilization of synergetic human- machine clouds: a big data cleaning case. In Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering, pages 15–18. ACM, 2014.

[12] Christian Keimel, Julian Habigt, Clemens Horch, and Klaus Diepold. Video quality evaluation in the cloud. In Packet Video Workshop (PV), 2012 19th International, pages 155–160. IEEE, 2012.

[13] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications, 1(1):7–18, 2010

[14] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1):50–55, 2008.

[15] Frank Leymann and Dieter Fritsch. Cloud computing: The next revolution in it. Proceedings of the 52th Photogrammetric Week, pages 3–12, 2009.

[16] Ling Qian, Zhiguo Luo, Yujian Du, and Leitao Guo. Cloud computing: An overview. In Cloud computing, pages 626–631. Springer, 2009.

[17] Steve Lohr. Google and ibm join in cloud computingresearch. New York Times,

[18] Mladen A Vouk. Cloud computing–issues, research and implementations. CIT. Journal of Computing and Information Technology, 16(4):235–246, 2008.

[19] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. Communications of the ACM, 53(4):50–58,2010

[20] Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, and Alfons Kemper. Capacity management and demand prediction for next generation data centers. In Web Services, 2007. ICWS 2007. IEEE International Conference on, pages 43–50. IEEE,

[21] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, and Pawan Goyal. Dy- namic provisioning of multi-tier internet applications. In Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on, pages 217–228. IEEE, 2005.

[22] Pradeep Padala, Kang G Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, Arif Merchant, and Kenneth Salem. Adaptive control of virtualized re- sources in utility computing environments. In ACM SIGOPS Operating Systems Review, volume 41, pages 289–302. ACM, 2007.

[23] Pankesh Patel, Ajith H Ranabahu, and Amit P Sheth. Service level agreement in cloud computing. 2009.

[24] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: issues and challenges. In Advanced Information Networking and Applications (AINA), 2010

[25] Allan Leinwand. The hidden cost of the cloud: Bandwidth charges, 2009.

[26] Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. Crowdsourcing systems on the world-wide web. Commun. ACM, 54:86–96, April 2011.

[27] Jeff. Howe. The rise of crowdsourcing. Wired, June 2006.

[28] Salman Ahmad, Alexis Battle, Zahan Malkani, and Sepander Kamvar. The jabberwocky programming environment for structured social computing. In Proceedings of the 24th annual ACM symposium on User interface software and technology, pages 53–64. ACM,2011.

[29] Maja Vukovic and Claudio Bartolini. Towards a research agenda for enterprise crowdsourcing. In Tiziana Margaria and Bernhard Steffen, editors, Leveraging Ap- plications of Formal Methods, Verification, and Validation, volume 6415 of Lecture Notes in Computer Science, pages 425–434. Springer Berlin / Heidelberg, 2010.

[30] M. Vukovic. Crowdsourcing for enterprises. In Services - I, 2009 World Conference on, pages 686 –692, july 2009.

[31] Natala J. Menezes Jenny J. Chen and Adam D. Bradley. Opportunities for crowdsourcing research on amazon mechanical turk. In Proceeding of The CHI 2011

[32] Alexander J. Quinn and Benjamin B. Bederson. Human computation: a survey and taxonomy of a growing field. In Proceedings of the 2011 annual conference on Human factors in computing systems, CHI '11, pages 1403–1412, New York, NY, USA, 2011. ACM.

[33] Mohammad Allahbakhsh, Boualem Benatallah, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Elisa Bertino, and Schahram Dustdar. Quality control in crowdsourcing systems: Issues and directions. Internet Computing, IEEE, 17(2):76–81, 2013.

[34] H. Amintoosi and S.S. Kanhere. A trust-based recruitment framework for multi-hop social participatory sensing. In Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on, pages 266–273, May 2013.

[35] Haleh Amintoosi, Salil Kanhere, and Mohammad Allahbakhsh. Trust and privacy considerations in participant selection for social participatory sensing. Technical Report UNSW-CSE-TR-201409, UNSW, 2014.

[36] Murat Demirbas, Murat Ali Bayir, Cuneyt Gurcan Akcora, Yavuz Selim Yilmaz, and Hakan Ferhatosmanoglu. Crowd-sourced sensing and collaboration using twit- ter. In World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010

[37] Michael S. Bernstein, Desney Tan, Greg Smith, Mary Czerwinski, and Eric Horvitz. Personalization via friendsourcing. ACM Trans. Comput.-Hum. Inter- act., 17(2):6:1–6:28, May 2008.

[38] Maja Vukovic, Mariana Lopez, and Jim Laredo. Peoplecloud for the globally in- tegrated enterprise. In Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops, volume 6275 of Lecture Notes in Computer Science, pages 109–114. Springer Berlin / Heidelberg, 2010.

[39] Anand P Kulkarni, Matthew Can, and Bjoern Hartmann. Turkomatic: automatic recursive task and workflow design for mechanical turk. In CHI'11 Extended Ab- stracts on Human Factors in Computing Systems, pages 2053–2058. ACM, 2011.

[40] Aniket Kittur, Susheel Khamkar, Paul Andr´e, and Robert Kraut. Crowdweaver: visually managing complex crowd work. In Proceedings of the ACM 2012

conference on Computer Supported Cooperative Work, pages 1033–1036. ACM, 2012.

[41] Steven Dow, Anand Kulkarni, Scott Klemmer, and Bjorn Hartmann. Shepherding the crowd yields better work. In Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, CSCW '12, pages 1013–1022, New York, NY, USA, 2012. ACM.

[42] Mohammad Allahbakhsh, Aleksandar Ignjatovic, Boualem Benatallah, Seyed-Mehdi-Reza Beheshti, Elisa Bertino, and Norman Foo. Reputation management in crowdsourcing systems. In Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on, pages 664–671. IEEE, 2012.

[43] Gang Wang, Christo Wilson, Xiaohan Zhao, Yibo Zhu, Manish Mohanlal, Haitao Zheng, and Ben Y Zhao. Serf and turf: crowdturfing for fun and profit. In Proceedings of the 21st international conference on World Wide Web, pages 679–688. ACM, 2012.

[44] Aniket Kittur Boris Smus Jim Laredoc Maja Vukovic Jakob Rogstadius, Vassilis Kostakos. An assessment of intrinsic and extrinsic motivation on task perfor- mance in crowdsourcing. In Proceeding of the Fifth International AAAI Conference on Weblogs and Social Media. AAAI, 2011.

[45] Joel Ross, Lilly Irani, M Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the crowdworkers?: shifting demographics in mechanical turk. In CHI'10 Extended Abstracts on Human Factors in Computing Systems, pages 2863–2872. ACM, 2010.

[46] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, and Thomas Sandholm. What's inside the cloud? an architectural map of the cloud landscape. In Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing, pages 23–31. IEEE Computer Society, 2009

[47] Georg Lackermair. Hybrid cloud architectures for the online commerce. Proceeding of Computer Science, 3:550–555, 2011. 1966.

[48] Sihem Amer-Yahia, AnHai Doan, Jon Kleinberg, Nick Koudas, and Michael Franklin. Crowds, clouds, and algorithms: exploring the human side of big data applications. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pages 1259–1260. ACM, 2010.

[49] Maja Vukovic. Crowdsourcing for enterprises. In Services-I, 2009 World Confer- ence on, pages 686–692. IEEE, 2009.

[50] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E Kraut. Crowdforge: Crowdsourcing complex work. In Proceedings of the 24th annual ACM symposium on User interface software and technology, pages 43–52. ACM, 2011.

[51] Greg Little, Lydia B Chilton, Max Goldman, and Robert C Miller. Turkit: tools for iterative tasks on mechanical turk. In Proceedings of the ACM SIGKDD workshop on human computation, pages 29–30. ACM, 2009.

[52] Michael S Bernstein, Greg Little, Robert C Miller, Bjorn Hartmann, Mark S Ack- erman, David R Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. In Proceedings of the 23nd annual ACM symposium on User interface software and technology, pages 313–322. ACM, 2010.

[53] Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In Proceedings of the 2011

[54] Adam Marcus, Eugene Wu, David R Karger, Samuel Madden, and Robert C Miller. Demonstration of qurk: a query processor for humanoperators. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data

[55] Abraham Bernstein, Mark Klein, and Thomas W Malone. Programming the global brain. Communications of the ACM, 55(5):41–43, 2012.

[56] Jeffrey Philip Bigham, Walter Stephen Lasecki, Kyle Ian Murray, and Samuel Christopher White. Closed-loop crowd control of existing interface, July 10 2012. US Patent App. 13/545,280.

[57] Norbert Wiener. The human use of human beings: Cybernetics and society. Number 20. Da Capo Press, 1954.

[58] William Ross Ashby et al. An introduction to cybernetics, volume 2. Chapman & Hall London, 1956.

[59] Joseph Carl Robnett Licklider. Man-computer symbiosis. Human Factors in Elec- tronics, IRE Transactions on, (1):4–11, 1960.

[60] Andrew Tanenbaum and Maarten Van Steen. Distributed systems. Pearson Pren- tice Hall, 2007.

[61] David Alan Grier. When computers were human. Princeton University Press, 2013.

[62] Saeed Arbabi, Mohsen Sharifi, Seyedeh Leili Mirtaheri, and Ehsan Mousavi Khaneghah. A low-overhead structure maintenance approach for building robust structured p2p systems. In Telecommunications (IST), 2012 Sixth International Symposium on, pages 586–591. IEEE, 2012.